Refinement for Ontology Evolution in Virtual Enterprises

Li Li, Baolin Wu, Yun Yang

CICEC Centre for Internet Computing and E-Commerce Faculty of ICT, Swinburne University of Technology
PO Box 218, Hawthorn, Melbourne, Australia 3122
{lli,bwu, yyang}@it.swin.edu.au

ABSTRACT

Virtual enterprise is based on the premise that work should be done where it can be done most optimally. In virtual enterprises, geographical boundaries merge seamlessly. It enables organisations to act in a way of flexibility and ability to adapt to rapid changes on the fly. However, different parties in a virtual enterprise must understand each other before they go further details in business. Ontologies are such kinds of ideal baselines to assist parties to communicate. One of the essential research issues with ontology is how to deal with changes during their evolving cycle. Therefore, ontology refinement is a crucial component in ontology evolution. This paper presents a taxonomy structure focusing on the is-a relations. In particular, the concept of closeness measurement is introduced based on the "distance" estimation. An extended cluster analysis process is provided. According to the algorithm presented, a new concept is generated according to its attributes. Additionally, the refinement mechanisms for primitive operations are proposed. Unlike some other ontology refinement mechanisms which leave ontology consistency checking to human users after modification, our method emphasises the importance of consistency checking by applying description logics which is demonstrated based on the proposed ontology.

Keywords: ontology evolution, virtual enterprises, clustering, description logic, consistency checking

1. INTRODUCTION

An ontology can be defined as *an explicit specification of conceptualisation* [6]. Currently, the use of ontology for different purposes such as Web applications, Web semantics, information systems, are intensively addressed. The ontology is important because it is the foundation for different organisations to understand each other within or across enterprises. Feasible ontologies in virtual enterprises (VEs) are especially important because of VE's characteristics: VEs only exist for a short life span as a temporary network of geographically distributed partners that cooperatively work together to share skills, costs, profits, risks, and markets, and, at the same time, decrease the investment [3]. VEs are not rigid organisational structures within rigid frameworks, but rather, heterogeneous ensembles, continuously evolving over time [13]. For this reason, a complete ontology is essential for VE formation because of its pressing time requirement and heterogeneity. However, there indeed exist cases when some concepts/attributes and their relations are out of date or cannot be accessed through the current ontology. According to Stojanovic [14], there are seven different requirements demanding ontology evolution. In addition, a six-phase evolution process is addressed. Other researchers [12,14] also note that ontology development is a dynamic process starting with an initial rough ontology, which is revised, refined and filled with details on an ongoing basis. That is to say, ontology refinement is important and needs more efforts throughout its entire lifecycle. In VEs where different parties might work on an ontology collaboratively, dynamically refining the ontology is likely to be more important than that in any other places.

Intelligent software agents (agents hereafter) are a suitable means of representing the partners of a VE [5]. Agents act on behalf of the enterprises or organisations that collaborate among themselves to achieve a specific goal under assumptions that they are goal-oriented, commitment-based and able to share skills, cost, profits, risks and markets [12]. We assume agents who are likely to form a VE have similar but slightly differing ontologies rather than completely different ones. This is reasonable in the real world for different partners having something in common within the domain.

Some work has been performed on tackling ontology refinement. Noy's work [10] talks about that both *merging* and *alignment* are two processes which are usually used to handle the ontology refinement. These two processes are illustrated in PROMPT [11], a semi-automatic approach to ontology merging and alignment where users' intervention is required. In his book [8], Maedche mentions an essential part of ontology engineering ontology learning. The development of the taxonomic backbone of the ontology is also involved. The clustering analysis mainly discussed in natural language processing previously, has been highlighted from learning the taxonomic relation perspective.

Generally speaking, existing refining proposals fall into two categories. One is to investigate ontology dynamic changes and the corresponding management from the knowledge engineering perspective [1,4,8,14]. The other is from the context of plan execution [9]. However, in light of their proposed approaches, a new term's relationships with existing concepts are specified/semi-specified by human users. By doing so they avoid indicating how they get to know the relations between con-

cepts. That is to say that they fail to address how they get to know the relations between new terms and the existing ones and whether the ontology is still consistent after some modifications.

To comply with the requirements, our main task will be on adjusting the proposed ontology structure to any changes of concepts and their relations while at the same time processing consistency checking with the source ontology. We use a statistics-based approach, especially the cluster analysis method first to obtain a corresponding classification of the terms (in the is-a concept hierarchy). Then we use description logics (DLs) [2,7] to check if the newly added concepts are consistent with the original ontology structure by considering their satisfiability and subsumption. The novel contribution of this paper is to provide a method that operates in a resilient manner in ontology refinement, and at meantime, consistency checking complying with the ontology evolution life cycle to guarantee that any application has a sound knowledge of the world.

We do not distinguish terms between parties, partners, participants and agents in this paper only if they would convoy much more meaning in certain circumstance.

The rest of this paper is organised as follows. The next section first presents a taxonomy structure and then addresses the mapping between concept relations and distances. Section 3 proposes refinement mechanisms for the add and delete primitive operations. Section 4 discusses consistency checking of an ontology by using DLs. Section 5 illustrates our approach with an example. Finally, section 6 concludes our work.

2. CLUSTER ANALYSIS

Term *cluster analysis* [15] actually encompasses a number of different classification algorithms. Two things are worth noting here. One general question facing *cluster analysis* in many areas is how to *organise* observed data into meaningful structures—taxonomies. Another is a suitable algorithm for a specific question. In this paper, concepts are organised as the higher the level of concept aggregation the more abstract these concepts are in the respective class. We borrow term "distance" to show close/loose relations in the paper. In addition, we use the term "new cluster" to notate a new generated concept if needed in a dendrogram. It will be discussed in more detail in section 3.

2.1 Taxonomy Structure

A concept definition, along with its relations and others (if they exist), is more likely to be described in a taxonomy (the "is-a" hierarchy) structure. The most central relations are the is-a relation (concept—concept) and property-of relation (concept—attribute). Like in Figure 1, nodes in squares with capital letters such as *A*, *B*, *C*, *D*, E, and F are abstract concepts whereas other nodes notated in circle with lowercases are concrete ones. The difference between an abstract concept and a concrete one lies in that the abstract one subsumes sub concepts. In this paper, concrete concepts are specifies with primitive or common knowledge in terms of attributes and properties. For instance, in a general "transport" ontology which describes different transports such as "ship", "airplane", "bus", and "train" etc. and their relationships, concepts like "ship", "airplane", "bus", and "train" are abstract concepts, while "mph", and "capacity" are concrete concepts to define the attributes of a specific transport. Clearly, concept *B* subsumes both *E* and *F*, whereas *B* is subsumed by another concept *A*. And lowercase f is a property-of concept *A*. Obviously, the higher the concept is, the more abstract it is. Ontology structure in this paper is described in the "is-a" hierarchical structure. And we will refer to Figure 1 again in subsequent sections. Understanding how a dendrogram is constructed, and how it should be interpreted, is helpful to understand ontology refinement by using clustering analysis. The next subsection will discuss Figure 1 in-depth from the cluster analysis perspective.

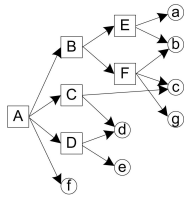


Fig. 1. Taxonomy structure of concepts

2.2 Mapping Concept Relations and Distances

Every concept, no matter it is a concrete or an abstract concept, has a set of attributes with different levels of granularity. Generally speaking, a concept can be expressed by a set of triples of (attribute, value, weight), namely $A_{att}=<(a_1, v_1, w_1), (a_2, v_2, w_2),\ldots ,(a_m, v_m, w_m)>$, where a_j is an attribute and v_j is the corresponding value. Here w_j ($j \in [1, n]$) is represented by a real number in range [0,1]) which is an estimation standing for to what extent an attribute getting a specific value (1 means matching perfectly, while others mean they can be approximately equal to those values). An instance such as (shape, circle, 0.8) means that a concept has attribute "shape" with value "circle", but not a 100 percent circle, the closeness estimation is only 0.8 instead of 1.

In order to reduce redundancy, a value matrix (*VM* in short), a corresponding attribute vector (*AV*), and its corresponding weight matrix (*WM*, in which 0 means corresponding attribute is *N/A*) are introduced below. It is easy to get a triple mentioned above such as (shape, circle, 0.8).

$$\begin{bmatrix} shape \\ colour \\ \cdots \\ size \end{bmatrix} \quad (VM)$$

$$\begin{bmatrix} circle & red & \cdots & N/A \\ square & green & \cdots & saml \\ \cdots & \cdots & \cdots & \cdots \\ N/A & green & \cdots & big \end{bmatrix} \quad (AV)$$

$$\begin{bmatrix} 0.8 & 1 & \cdots & 0 \\ 1 & 0.6 & \cdots & 0.1 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0.9 & \cdots & 0.8 \end{bmatrix} \quad (WM)$$

It is clear that the concept similarity, which is based on

relevant matrixes and closeness measurement, determines the relations between the concepts. In terms of the closeness measurement, "distances" between the attributes will be calculated. Obviously, the smaller "distance" means much more similar than that of greater "distance".

In order to calculate the distance, similarity matrixes (also symmetric matrixes, notated as SM_1, SM_2, SM_j, …, SM_m for simplicity), where SM_j corresponds to the j^{th} attribute in the *AV* vector and their different values, are provided. These matrixes show how close these values are (similarity closeness ∈ [0, 1], in which 1 means perfectly matching, 0 means never matching, other value ∈ (0, 1) means approximate matching).

Let us recall all of them before we start to discuss our approaches. In order for agents to automatically refine their individual ontologies, firstly, a value matrix (*VM*), a corresponding attribute vector (*AV*), and its corresponding weight matrix (*WM*) must provide. Moreover we suppose that similarity matrixes (SM_1, SM_2, SM_j, …, SM_m) are available for the proposed ontology and their attributes. In addition, we always treat ontologies are consistent initially. Without this our consistency checking will make no sense.

3. REFINEMENT MECHANISMS

As indicated earlier, ontology evolves overtime in the real world. How to refine ontology on the fly needs to be handled properly. In this section, we consider two primitive operations, i.e. add and delete as the refinement mechanisms based on the ontology structure and cluster analysis discussed earlier in this paper.

3.1 Add Operation

When we consider the add operation, generally there are two cases: (1) a new concept is subsumed by another concept; (2) a new cluster (concept) needs to be created in order to subsume a new concept. It is also possible that a new cluster may generate another bigger new cluster afterwards. Let us consider new concept *X* (only considering an abstract concept here) in Figure 2. In one case (on the left), *X* is subsumed by concept *B*, while in another case (on the right) shows that a new cluster *Y* (being subsumed by concept *A*), is generated which subsumes both *B* and *X*.

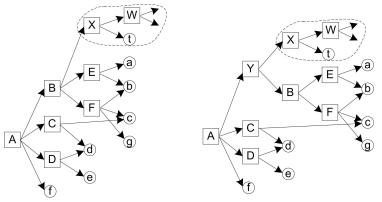


Fig. 2. Two cases of adding operation

For the reason that the number of the attributes might vary from concept to concept, for simplicity, the biggest number will determine the cardinality of the *VA*. In this case, there must be some empty elements in *VM*, here "*N/A*" is used to notate no such a property in *VM*. Figure 3 is the flow chart to add a new concept through top-down in the hierarchy.

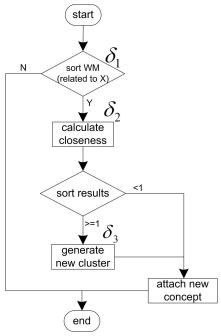


Fig. 3. Flow chart of adding a new concept

First of all, we assume that new concept *X* has the same attribute vector (*AV*) as others do. Of course, some properties are "*N/A*" under certain circumstance, so does the weight matrix (*WM*) with corresponding 0's value. Additionally, *X*'s weight vector X_w and value vector X_v are also provided. The details are as follows.

Step 1: **Function** *sort_WM*
It mainly deals with sorting weight matrix (*WM*) by referring to *X*'s 1's distribution by referring to its weight vector X_w (each $w_{Xj} \in [0, 1]$). The summary $\sum_{j=1}^{m} w_{ij} \cdot w_{Xj}$, where $i \in [1,n]$, $j \in [1, m]$, $w_{ij} \in [0,1]$, corresponding to the element in WM, $w_{Xj} \in [0,1]$, corresponding to the element in X_w is filtered by a threshold (d_1) which is given apriori.

(1) The values that are less than the threshold are ignored.
(2) If all values are less than the given value, the process leads to the *END* of the adding process. That is to say, the new concept *X* is irrelevant to the existing concept taxonomy structure.
(3) Otherwise all selected vectors form a new weight matrix *New_WM* and a new value matrix *New_VM* respectively (only related concepts are left after this step).

Step 2: **Function** *calculate_closeness*
As mentioned before (Section 2.2), SM_j (j∈ [1,*m*]) as a similarity matrix (it is symmetric) provides similarity measurements of different values upon a specific attribute. For example, an attribute "*shape*" has a matrix (SM_j) to estimate to what extent the shape "circle" and "ellipse" can be regarded as similar to each other. Figure 4 gives a demonstration. For new concept $X < w_{X1}, w_{X2}, \cdots, w_{Xm} >$, calculate $\sum_{j=1}^{m} w_{xj} \cdot s_j'$

where s_j' coming from above SM_j (j∈ [1,m]) (Figure 4) based on the *New_WM* (filtered).

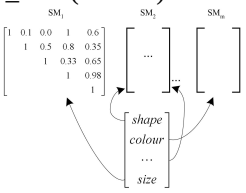


Fig. 4. Similarity matrixes

Step 3: **Function** *sort_result*
Sorting above values in the descending order and selecting candidates (notated as *Selected_New_WM*) whose summary values are greater than the threshold (d_2). If there is no one with a sum value greater than d_2, go to step 5. Again, getting corresponding *Selecte_New_VM* as well.

Step 4: **Function** *generate_new_cluster*
(1) Appending new concept *X*'s weight vector X_w and value vector X_v to the *Selected_New_WM* and *Selected_New_VM* respectively.
(2) Selecting all attributes (mapping to *AV* and satisfying corresponding elements in *Selected_New_WM* which are greater than threshold d_3) to form a *New_AV*.
(3) Making an intersection to obtain how many attributes are involved (with a corresponding weight in *Selected_New_WM* which is greater than d_3) if possible. Otherwise go to step 5.
(4) Making an intersection to obtain what values are involved (with a corresponding weight in *Selected_New_WM* which is greater than d_3) if possible. Otherwise go to step 5.
(5) The newly generated cluster features with value vector Y_v and weight vector Y_w as well. It is worthy noting that the newly generated cluster features with 1's (if it has corresponding attribute) or 0's (no such property) rather than other values between (0, 1).

Step 5: **Function** *attach_new_concept*
Attaching the newly generated concept to the original structure, such as in Figure 2, new concept *X* (enclosed in a dotted line circle) is attached to original node *B* if there is no new cluster generated, on the contrary, *Y* should be attached to node *A* if new cluster *Y* is generated while the new concept *X* is added to the structure.

3.2 Delete Operation

As for deleting a concept from the structure, generally, it should be relatively easy if a simple case (without any sub concepts) is concerned (Figure 5). For other cases, sub concepts of the deleted concept (subgraph) will be attached to its parents (Figure 6).

(1) In Figure 5, suppose the deleted concept is *t* (on the left), just delete it (see the graph on the right as the result).

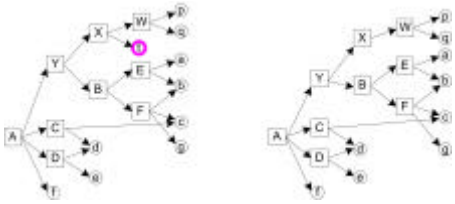


Fig. 5. Delete operation—case 1

(2) In Figure 6, suppose the deleted concept is *Y* (on the left) this time, all its (*Y*'s) sub concepts (both *X* and *B*) will be attached to its parent *A* (see the graph on the right as the result).

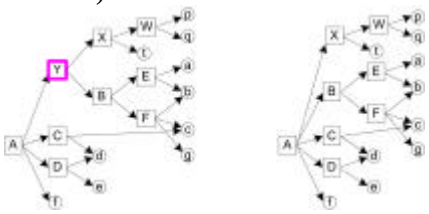


Fig. 6. Delete operation—case 2

Functions such as *generated_new_cluster* and *attach_new_concept* are very useful to explain the add and delete operations. We believe more complex structure can also get some help from approaches here.

4. CONSISTENCY CHECKING

Different partners can contribute to the extension of an ontology. In addition, the same partner may revise the ontology from time to time. Therefore, ontology consistency checking is vital to any applications based on it. Description logics (DLs) as carefully selected parts of first order predicate logics are designed to be expressive enough to be useful for constructing and querying ontologies. Their inference engines are good at answering subsumption and satisfiability queries. In the paper, we use them to check consistency of refined ontology. The basic ideas look as follows.

Generally, *C* and *D* denote any concepts, *R* for the role name. The semantics is given by means of interpretation *I*. An interpretation is a pair of domain Δ^I and interpretation function $\cdot^I$, namely $I = (\Delta^I, \cdot^I)$. With parts of constructors, interpretation *I* for concepts is defined in Table 1

Table 1. Parts of constructors of DLs

Constructor	Syntax	Semantics
Atomic role	R	$R^I \subseteq \Delta^I \times \Delta^I$
Conjunction (AND)	$C? D$	$C^I \bigcap D^I$
Disjunction (OR)	$C? D$	$C^I \bigcup D^I$
Negation (NOT)	$\neg C$	$\Delta^I \setminus C^I$

Existential restrict	$?R.C$	$\{x \mid \exists y \cdot < x, y > \in R^I \wedge y \in C^I\}$
Universal restrict	$\forall R.C$	$\{x \mid \forall y \cdot < x, y > \in R^I \Rightarrow y \in C^I$

In terms of ontology consistency checking, we have axioms like *C*? *D* (*C* is subsumed by *D*, "? " notates subsumption relation in DLs) iff *C*? ¬*D*? ⊥, that is to say *C*? ¬*D* is not satisfiable (inconsistent). We use annotations like *property1*, *property2*, etc. to notate different attributes shown in Figure 1 by ignoring what it really means. Based on the DL model, the taxonomy structure (Figure 1) can be expressed more formally as:

A? ∃*R*? *C*? *D*? ? *property1.f*

B? *A*

B? ∃*E*? *F*

C? *A*? ? *property2.c*? ? *property3.d*

D? *A*? ? *property3.d*? ? *property4.e*

E? *B*? ? *property5.a*? ? *property6.b*

F? *B*? ? *property6.b*? ? *property2.c*? ? *property7.g*

For new concept *X* from Figure 5, it looks like:

X? *B*

X? ∃*W*? ? *property7.t*

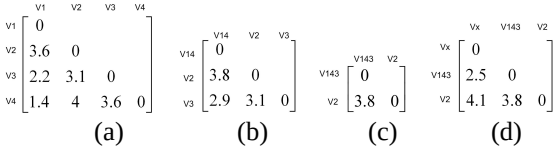
The satisfiability and subsumption can be used during ontology construction to verify whether the ontology is consistent or not. We assume it is consistent initially. When the primitive operations have been processed, DL can answer the questions such as 'is a concept consistent with the source ontology?', and 'Is this concept a part of another?'. For instance (Now look at the case that *X* is attached to *B*), *X*? ¬? *property1.f* is inconsistent because of *X*? *B* (given condition), *B*? *A* (already known condition) and which implies *X*? *A*. *X*? ¬? *property1.f* is not satisfiable because it contradicts with the reasoning result *X*? *A* (*property1.f* is one of properties of concept *A* shown in Figure 1). As the above consistency checking is conducted according to Figure 1 which is platform- and description language-independent, it is promising to deal with ontology refinement in the is-a hierarchy in general.

5. EXAMPLE

A simple example below will show how a new node is added and a new node is generated by applying cluster analysis. In this example, we assume d_1, d_2, and d_3 (Figure 3) equal to zero. Euclidean distance, computed as: $\text{distance}(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$, is used to measure "distance", and the shortest distance in a Euclidean distance matrix will determine which pair of nodes will be fused to form a cluster. First we demonstrate a dendrogram construction through the example, then we consider adding a new node to this hierarchical structure to refine it.

Now suppose there are 4 nodes V_1(2,3), V_2(5,1), V_3(4,4), V_4(1,2). Their corresponding Euclidean distance matrix is shown in Figure 7.a. It shows that the most similar nodes are V_1 and V_4 (the value is 1.4). Then we get a new cluster V_{14}(1.5,2.5), a means of these two nodes is calculated for each dimension. That is to say, after V_1 and V_4 clustered, a new node V_{14}(1.5,2.5) is generated. So now there are three nodes V_{14}(1.5,2.5), V_2(5,1), V_3(4,4). Repeat the above calculation, a revised distance matrix is shown in Figure 7.b. At this stage, the most similar nodes are V_{14} and V_3 (the value is 2.9). Then we get a new cluster, V_{143}(3.5,4.5). Now there are two nodes, V_{143}(3.5,4.5), V_2(5,1). Again we get a revised distance matrix which is shown in Figure 7.c.

The whole process is summarised by a dendrogram as shown in Figure 8 (on the left). Suppose a new node notated as V_x(6,5) is going to be added. As we mentioned before, the entire process works in a top-down approach. The same is here. However, in this example, we only focus on nodes on the first level (the root level namely the top level normally is virtual item annotated as "ENTIRY" or "THING". In real cases, the attribute matrix will determine to which level the algorithm is affectted. The worst case is that all existing concepts will be taken into consideration), namely V_{143}(3.5,4.5), V_2(5,1). At this stage, calculations are between these three nodes. That is V_x(6,5), V_{143}(3.5,4.5), and V_2(5,1). At the end, the distance matrix is shown in Figure 7.d. It is clear that the most similar pair is V_{143} and V_x (the value is 2.5). The revised dendrogram is shown in Figure 8 (on the right). **N**ew node V_x is added in and fused with V_{143} to form a cluster V_{x143} which can be regarded as a new generated concept in the ontology refinement process.



(a) (b) (c) (d)

Fig 7. Euclidean distance matrixes

This example demonstrates our ontology refinement process, especially the add operation works well by taking attribute-value pair "distance" and clustering analysis. It is no doubt that in the real world, the refinement for ontology, which includes many concepts, is definitely more complicated than the example. However, the basic ideas and mechanisms to refine the ontology are quite similar to the method here.

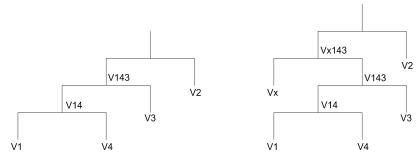


Fig. 8 New cluster in dendrogram construction

6. CONCLUSIONS AND FUTURE WORK

In this paper, our general focus is on developing a new approach to deal with the refinement for ontology evolution in virtual enterprises. A novel clustering algorithm has been applied to classify new concepts. Unlike other methods in ontology refinement, our approaches can classify the new concepts automatically instead of specified or semi-specified manually. In addition, the primitive add and delete operations are provided. Furthermore, we have illustrated that consistency checking should be a crucial element within the ontology evolution life cycle to guarantee that any application has a sound knowledge of the world. The consistency checking of the concepts makes our approach in accordance with knowledge management requirements in preserving knowledge and sharing knowledge. In addition, we believe that clustering analysis is a promising way to knowledge acquisition and ontology management. Moreover, differentiating abstract concepts from concrete ones are useful when we address the hierarchical taxonomic structures.

There are many issues remaining as future work. The semantic meaning of new clustering cannot be fully achieved only through the vector descriptions. Moreover, deleting is not as easy as people first thought. Another aspect is the structure for ontology consistency checking. Consistency checking needs to be emphasised in ontology evolution by considering multiple users. In addition, it is nevertheless essential to address the attribute weighting issue not only in a deterministic environment but also in a non-deterministic environment.

ACKNOWLEDGEMENT

Work reported in this paper is partly supported by Swinburne Vice Chancellor's Strategic Research Initiative Grant 2002-2004 for project "Internet-based e-business ventures".

REFERENCES

[1] Alfonseca, E. and Manandhar, S., Proposal for Evaluating Ontology Refinement Methods, *In: Proceedings of 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Spain, 2002. http://www.ii.uam.es/~ealfon/esp/pubs.html.

[2] Baader, F., Horrocks, I., and Sattler, U., Description Logics as Ontology Languages for the Semantic Web, *Lecture Notes in Artificial Intelligence. Springer-Verlag*, 2003. http://www.cs.man.ac.uk/~horrocks/Publications/.

[3] Byrne, J. A., Brandt, R., and Bort, O., The Virtual Corporation, *Business Week*, vol. 8, pp. 36-40, February 1993.

[4] Devedzic, V., Understanding Ontological Engineering, *Communications of the ACM*, 45(4), pp. 136-144, April 2002.

[5] Fischer, K., Müller, P. J., Heimig, I., and Scheer, W. A., Intelligent Agents in Virtual Enterprises, *In: Proceedings of the 1st International Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM'96)*, London, 1996.

[6] Gruber, T. R., Toward Principles for the Design of Ontologies Used for Knowledge Sharing, KSL-93-04, Knowledge Systems Laboratory, Stanford University. http://ksl-web.stanford.edu/.

[7] Horrocks, I. and Sattler, U., A Description Logic with Transitive and Inverse Roles and Role Hierarchies, *Journal of Logic and Computation*, 9(3), pp. 385-410, 1999.

[8] Maedche, A. Ontology Learning for the Semantic Web, Kluwer Academic Publishers, 2002.

[9] McNeill, F., Bundy, A., and Schorlemmer, M., Dynamic Ontology Refinement, Informatics Research Report EDI-INF-RR-0177, School of Informatics, University of Edinburgh, June 2003.

[10] Noy, N. F. and Klein, M., Ontology Evolution: Not the Same as Schema Evolution, *Knowledge and Information Systems*, In press. Available as SMI technical report SMI-2002-0926 (2002). http://smi.stanford.edu/people/noy/publications.html.

[11] Noy, N. F. and Musen, M. A., PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment, *In:Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, 2000.

[12] Petersen, S. A., Using Agents to Support the Selection of Virtual Enterprise Teams, *In: Proceedings of 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)* (at AAMAS 2002), Bologne, Italy, July 2002.

[13] Petersen, S. A., Divitini, M., and Matskin, M., An Agent-based Approach to Modelling Virtual Enterprises, *International Journal of Production Planning and Control*, 12(3), pp. 224-233, April 2001.

[14] Stojanovic, L., Maedche, A., Motik , B., and Stojanovic, N., User-driven Ontology Evolution Management, *In: Proceedings* of *the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW*, Madrid, Spain, 2002.

[15] Tryon, R. C. and Bailey, D. E. *Cluster Analysis*, McGraw-Hill, New York, NY, 1973.